Fermilab Experience with OKD (OpenShift)

Anthony Tiradani

K8s-HEP Meetup

01 December 2020

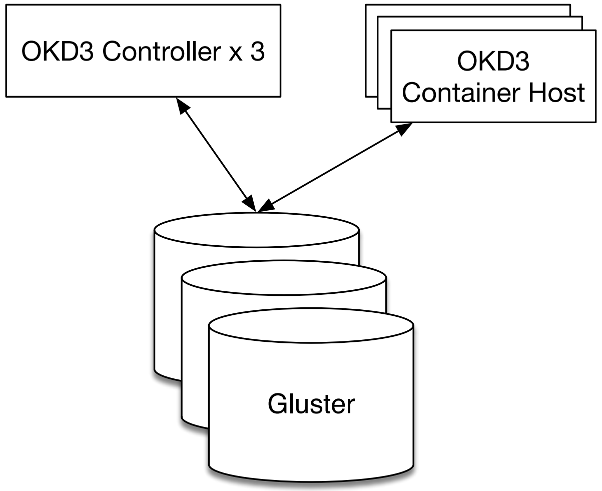
Disclaimer and Acknowledgements

- I am presenting a lot of work others have done. All credit goes to them!
- I am focusing on Fermilab specific experiences with OKD

- A not exhaustive list of people who should get credit are:
  - Maria Acosta, Nick Smith, Lindsey Grey
  - Chris Bonnaud, Ed Simmonds, Glenn Cooper
  - Farrukh Khan, Burt Holzman

OKD Infrastructure

- Private Docker-like registry – Harbor [https://goharbor.io/]
- Current deployed OKD version is 3.11
  - CMS Development cluster
  - FNAL Production cluster
  - Backed by Gluster storage for persistent volumes
- Future plans
  - Upgrade to OKD4
  - Move from Gluster to Ceph (mandated by OKD4)
  - Provide a development platform as well as managed production cluster(s)



OKD Security Model in the Context of SLATE

- Administrators are generally uncomfortable turning over the k8s keys to users off the street (e.g. a remote team that deploys edge services for you)
- It’s easy to deploy a less secure k8s than desired or intended
  - ecosystem is getting better
  - a good chunk of the RBAC code was contributed by RH
- The OKD security model has some nice features like:
  - non-root containers by default (and is suggested as best practice)
  - access-based rules that isolate user namespaces from each other

The downside of course is that images and operators off the street may not work without some changes. That could be collaborative work.

CVMFS “Problem”

- Several approaches to providing CVMFS:
  - Cvmfsexec [https://github.com/cvmfs/cvmfsexec]
    - Needs privileged pods for older kernels (kernels < 4.18 on RHEL8 or < 3.10.0-1127 on RHEL 7.8)
    - Privileged pods running user jobs are a no-go for us and FNAL Cybersecurity
  - CSI-CVMFS [https://gitlab.cern.ch/cloud-infrastructure/cvmfs-csi]
    - Good, “compliant” API wise approach. In use at CERN with Ceph
    - Designed for vanilla Kubernetes, we tried porting it to Openshift but:
      - Feature is in “Technology Preview”, not meant for production as of OKD3.11
      - Will revisit in OKD4
    - Needs a substantial level of privilege including bidirectional host mounts and DaemonSet privileged pods in all hosts.
    - API server and Kubelet need to be started with the --allow-privileged=true flag, cluster is being shared by multiple people, allowing privileged widely, not something we can do.

CVMFS “Problem” (Continued)

- Several approaches to providing CVMFS:
  - Parrot plugin
    - Needs **ptrace** enabled, NSA advices against this [NSA Recomendation document] not tested yet

[AND MANY MORE]

(Providing CVMFS is subjective to each setup, no wrong answer, just different)

CVMFS "Problem" - Our Solution

We want no privileged pods running user jobs – what to do?

- Back to basics – NFS
- Inevitable trade-off performance vs. security (We MUST comply with DOE,FNAL, etc.)

Pros

- Kubernetes maintains a seamless NFS integration out of the box
- Two ways of doing it:
  - **Ephemeral NFS volume**: works as a normal NFS mount, you just need an IP/Hostname of a server.
  - **Persistent Volume with NFS**: K8s Provisions a managed resource (PV) within the cluster which is accessed via NFS. Persistent (no need to reload data or cache every time a pod/container mounts it) + can be shared between multiple pods
- Doesn't need Privileges!

Cons

- Needs a robust underlying storage AKA large Hardware, not easy to get these days
- Introduces a single point of failure and a bottleneck
- Need to set up and manage a Baremetal NFS server?

Fermilab

CVMFS “Problem” – Our Solution (Continued)

- Kubernetes has mechanisms in place such that it will always maintain the state and QoS an application needs
- Autoscaling and health probes are constantly monitoring load and traffic to/from pods
- First test with external machine was very slow, couldn’t keep up with a decent amount of load

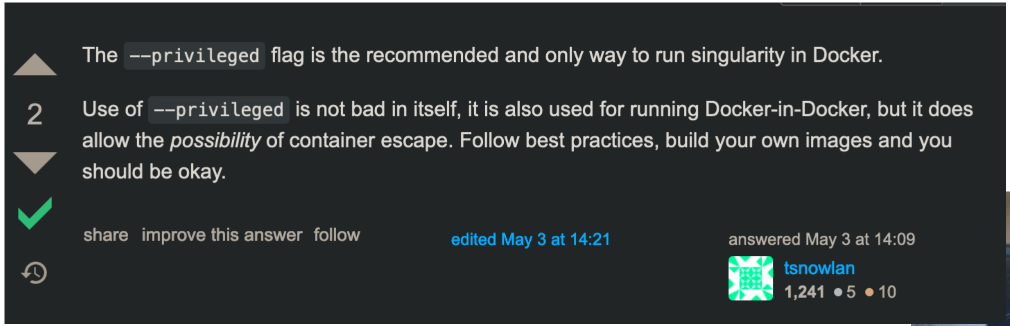
kubernetes OPENSHIFT

jupyterhub for kubernetes

Our approach so far:

- Move the NFS server away from bare metal and let Kubernetes do the work
- Autoscaling increases or decreases the number of servers behind a load balancer depending on load
- Health probes detect problematic servers and act
- (remove from LB, kill bad server pod, fire up a new server pod) in matter of seconds
- Allows us to provision custom Jupyter notebooks with CVMFS
  - ** Still needs privileges but pods are invisible and inaccessible to users running condor/dask/spark jobs**

OKD and Singularity



2

The `--privileged` flag is the recommended and only way to run singularity in Docker.

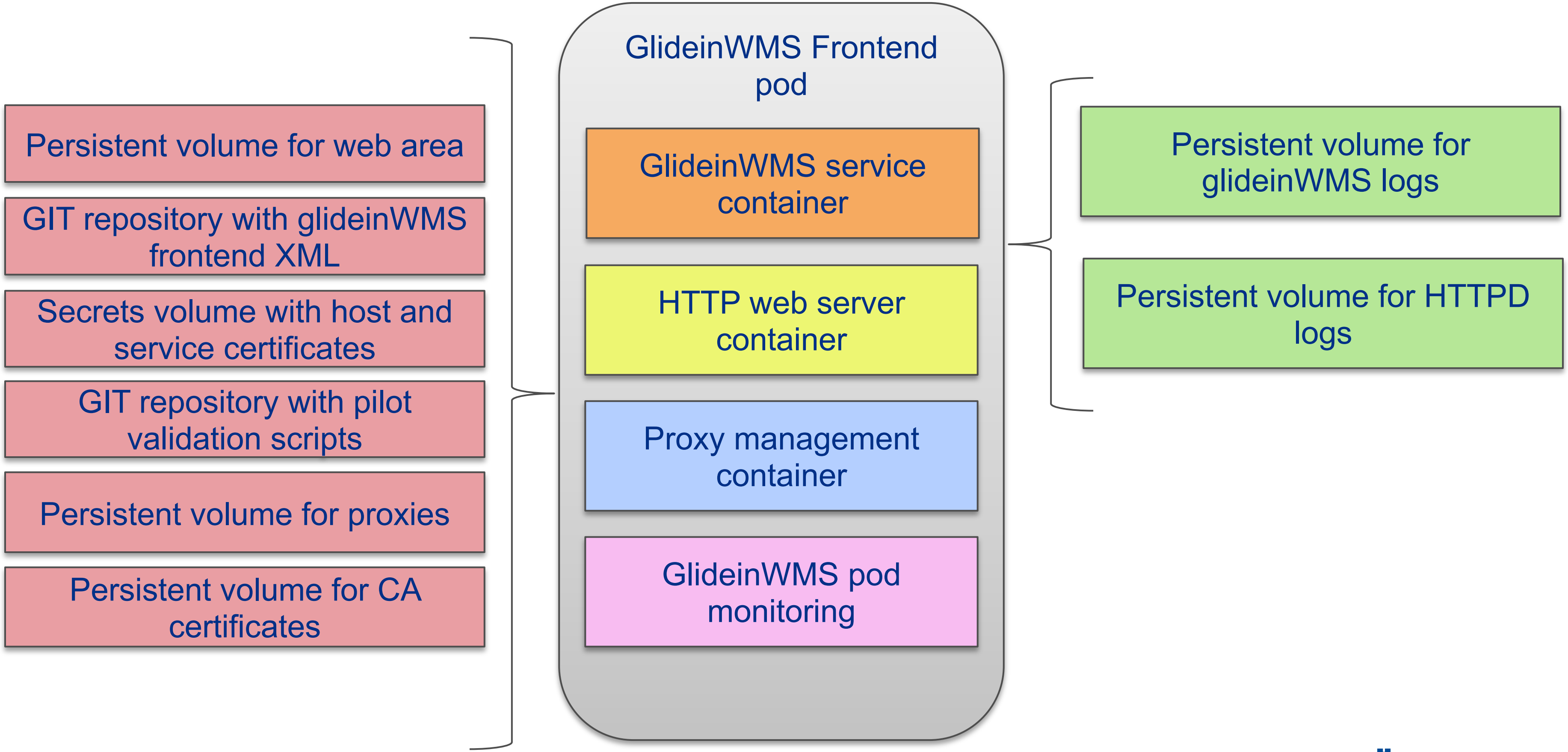
Use of `--privileged` is not bad in itself, it is also used for running Docker-in-Docker, but it does allow the *possibility* of container escape. Follow best practices, build your own images and you should be okay.

- Singularity <u>will not</u> run within unprivileged Docker containers!
- Suggestions from the audience welcome and appreciated!

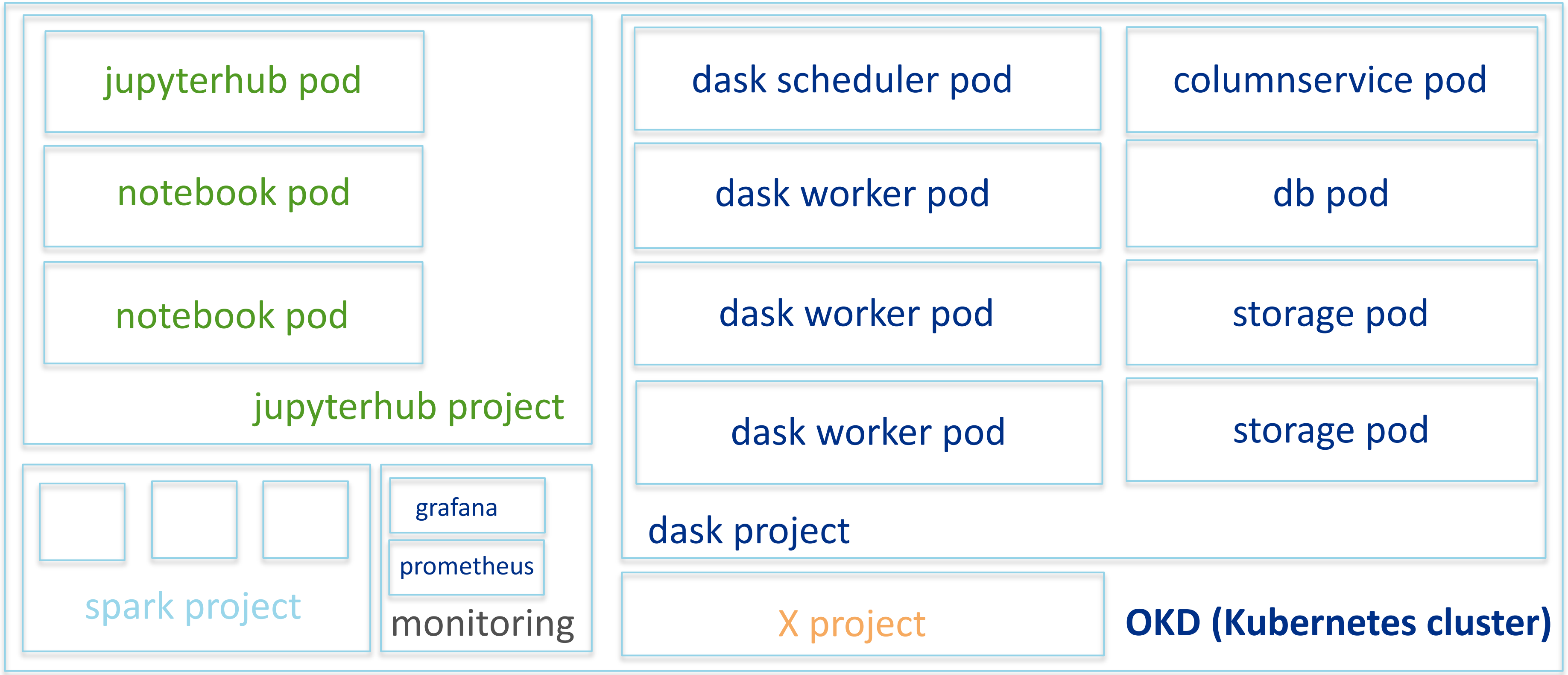
HTCondor

- HTCondor StartD
  - Must run privileged for Singularity to work (see previous slide)
  - Not in production
    - Security hurdles to cross due to privileged mode requirements
    - Still testing corner cases
  - Looking forward to the K8s integration that HTCondor team is working on
  - Open question: Does HTCondor handle scheduling, or do we have Kubernetes handle scheduling?

GlideinWMS Frontend


Persistent volume for web area
GIT repository with glideinWMS frontend XML
Secrets volume with host and service certificates
GIT repository with pilot validation scripts
Persistent volume for proxies
Persistent volume for CA certificates
GlideinWMS Frontend pod
GlideinWMS service container
HTTP web server container
Proxy management container
GlideinWMS pod monitoring
Persistent volume for glideinWMS logs
Persistent volume for HTTPD logs

Elastic Analysis Facility



Discussion

Questions?

Looking forward to hearing others experiences and solutions to problems we've encountered!

Fermilab